Einstieg in die Implementierung

MEA, Workshop 3, 29.10.2026

Der Tag

10:00	Ankommen, Rückblick mit dem Tutor, Überblick
10:15	Walking Skeleton: der reduzierte Ablauf der Fallstudie
10:45	AsyncAPI live
11:15	Teams zeigen ihre Contracts am Bildschirm
12:00	Mittagspause
12:45	Testen des Event-Katalogs als Gruppen-Übung
14:30	Welche Clients gibts?
15:45	Einstieg ins Coding - entwickeln und testen ohne Infrastruktur
16:45	Ausblick und Hausaufgabe

Rückblick auf den 22.10. mit dem Tutor

- Tutor-Prompt für den Rückblick: archi-lab.io/tutor/mea_ws26/2026-10-22-rueckblick.txt
- auch auf der Kursseite, Workshop 29.10., Agenda 10:00
- alles kopieren, in die eigene KI einfügen
- der Tutor fragt nacheinander:
  - Was haben wir letztes Mal gemacht?
  - Was hat dir besonders gut gefallen?
  - Was hat dir nicht gefallen?
  - Worüber sollten wir noch einmal weiter reden?

> Ca. 10 Minuten - und am Ende Learnlog-Eintrag generieren lassen!

Walking Skeleton: was ist drin?

Die Fallstudie gibt es in drei Fassungen, alle in der Dokumentation:

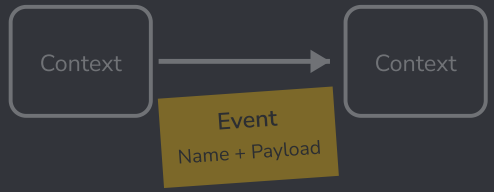
- Fallstudie im Original: der Text, wie er im EventStorming galt
- Fallstudie: mit den Änderungen seit dem EventStorming, gelb markiert
- Fallstudie für das Walking Skeleton: was im Walking Skeleton dabei ist, vereinfacht wird oder fehlt

Der Happy Path

1. Gast scannt den QR-Code am Sitzplatz, schaut durch die Speisekarte und bestellt
2. Die Bestellung geht an die bemannte Station, die Rezeptschritte erscheinen
3. Die Schritte werden nacheinander bestätigt, nach dem letzten wird ein Fach zugewiesen
4. Das Essen liegt im Fach, die Bestellung ist fertig, der Gast bekommt die Fachnummer
5. Der Gast wählt das Fach an der Abholstation, das Fach öffnet sich
6. Das Fach ist wieder frei, die verbrauchten Zutaten werden für den Tagesbericht gezählt

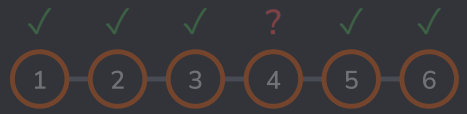
Testen des Event-Katalogs als Gruppen-Übung

1 Die Grenzen



Wir gehen zur Context Map. Zu jedem Pfeil legen wir Name und Payload des Events fest und kleben es als Sticky aufs Whiteboard. Ein Event ist ein Fakt, kein verstecktes Kommando.

2 Der Happy Path



Ist jeder Schritt durch ein Event gedeckt?

Die ganze Gruppe geht den Happy Path Schritt für Schritt durch. Das Team, das ein Event erzeugt, zeigt seinen Contract. Die Teams, die es nutzen, prüfen: Kann ich damit arbeiten? Fehlen Daten?

3 Die Probe



Wir gehen die synchronen Calls aus ST2 durch. Für jeden entscheiden wir: Haben wir die Events, die ihn ersetzen?

Was offen bleibt, kommt auf eine gemeinsame Liste, mit Namen und Termin.

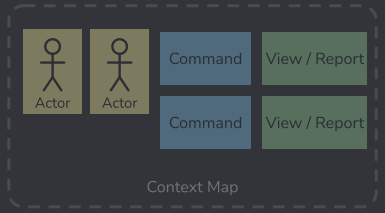
Die Probe: synchrone Calls aus ST2

Order → CookingStation	beim Bestellen: mindestens eine Station online?
CookingStation → Order	Koch meldet fertig: Bestellung abholbereit
Order → Menu → Order	Order kennt die Gerichte // Beliebtheit: wie oft an Tag X bestellt?
Order → CookingStation	Dispatch: freie Station übernimmt die Bestellung
Table → Order → Table	Gäste können nicht auschecken, wenn noch Bestellung offen
Menu → Order	Gericht aus aktiver Bestellung kann nicht von Karte genommen werden

> Best-Of aus Ihren Karten und meiner eigenen Einschätzung - gern ergänzen!

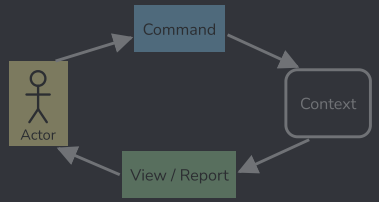
Clients: welche gibt es, und wer ist zuständig?

1 Actors und UIs



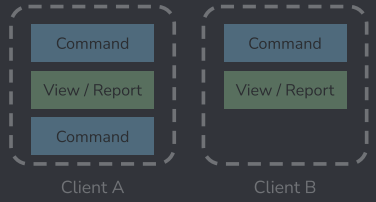
Actors, Commands und Views kommen auf die Context Map: Gast, Köchin oder Koch, Restaurant Manager, Guestroom Manager. Blau ist, was sie auslösen, grün (View / Report), was sie sehen, zum Beispiel der Tagesbericht der verbrauchten Zutaten.

2 Pfeile



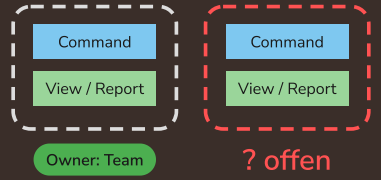
Pfeile zeigen die Richtung: vom Actor zum Command und weiter zum Context, vom Context zur View und weiter zum Actor. Jedes Team zeichnet die Pfeile an seinem Context.

3 Clients bilden



Commands und Views, die zusammengehören, bilden einen Client. Offen ist zum Beispiel: Teilen sich beide Manager einen Admin Client, oder bekommt jede Rolle einen eigenen?

4 Zusagen



Zwei Listen: Client gesucht und Client-Owner. Ein Client geht an das Team, dessen Context die meisten Daten liefert. Was niemand übernimmt, bleibt sichtbar auf der Liste.

Damit sollten wir dann alle Clients beisammen haben.

Die einfachste Client-Architektur

- eine Hülle, die Bereiche pro Client, getrennt (Tabs oder Routen)
- keine Integration zwischen den Bereichen
- jeder Bereich spricht nur mit seinem Backend
- Backend for Frontend: am 05.11.

Prinzipien für Microservices

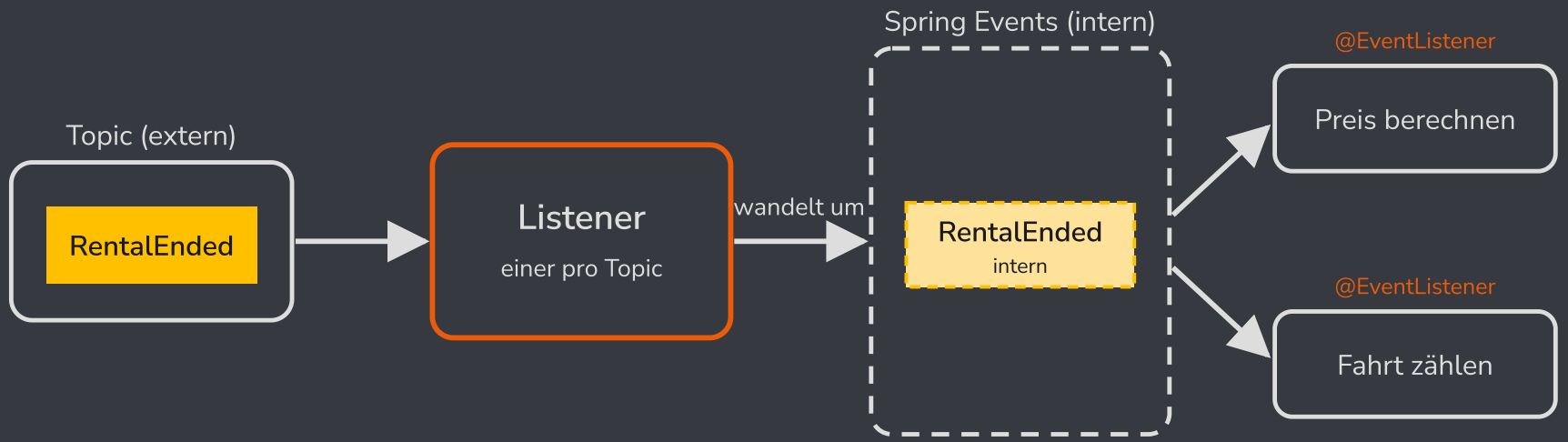
- 1 Bounded Context = 1 Service = 1 Team
- lieber zu groß anfangen, später teilen
- Aggregate = Einheit der Konsistenz im Service
- Event = ein Aggregate hat sich geändert
- Kommunikation vor allem asynchron, über Events
- Transport: Kafka
- Konventionen + Spezifikation pro Event: Event-Contract

Ein Service pro Bounded Context als Startpunkt: Vaughn Vernon, berichtet in InfoQ, 31.07.2016 (infoq.com/news/2016/07/microservices-ddd-vernon); Microsoft, .NET Microservices Architecture, Kapitel "Identifying domain-model boundaries". Grenzen zwischen Services sind schwer zu verschieben: Martin Fowler, MonolithFirst, 03.06.2015 (martinfowler.com/bliki/MonolithFirst.html).

Entwickeln und testen ohne Infrastruktur

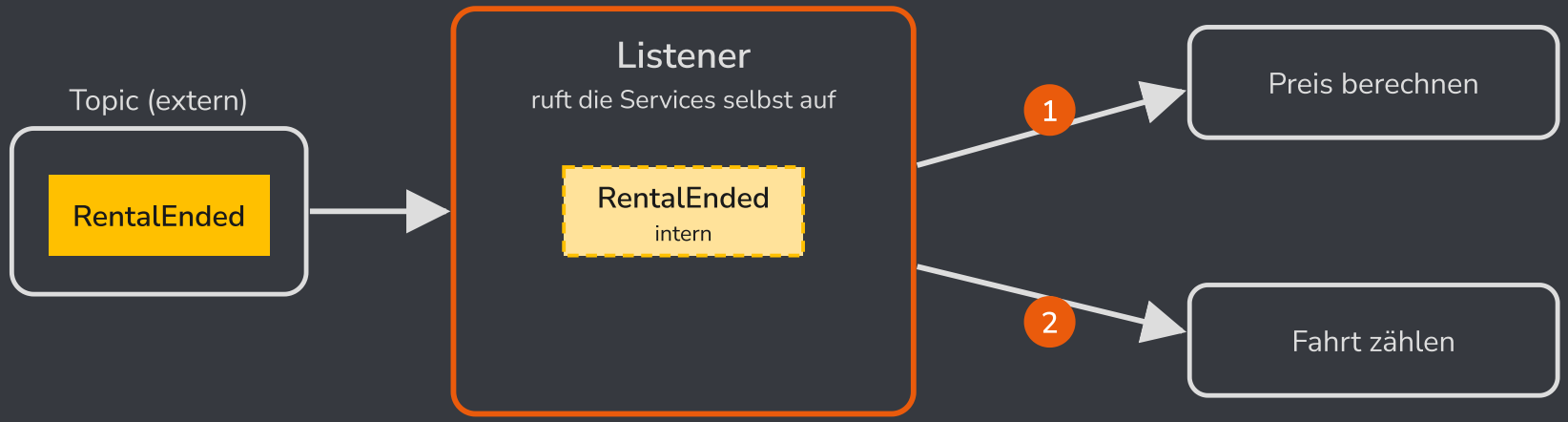
- der Service läuft allein, ohne Kafka und ohne die Nachbarn
- Eingänge: REST und Event-Consumer, Ausgang: ein Port zum Veröffentlichen von Events
- Tests gegen den Application-Layer, mit einem Fake für das Veröffentlichen
- Events der Nachbarn von Hand einspeisen

Ein Listener pro Topic, Variante 1: Spring Events



Vorteil	der Listener weiß nicht, wer reagiert: eine neue Reaktion ist eine neue Klasse
Preis	kein Ort, an dem alle Reaktionen und ihre Reihenfolge zu sehen sind

Ein Listener pro Topic, Variante 2: Orchestrator



Vorteil	alle Reaktionen und ihre Reihenfolge an einer Stelle, im Debugger Schritt für Schritt, kein Spring nötig
Preis	der Listener kennt jede Reaktion: eine neue Reaktion ändert ihn

Womit Sie jetzt anfangen können

- Repository klonen, Projektgerüst anlegen
- den Happy-Path-Ausschnitt Ihres Bounded Context als Test beschreiben
- Events passend zum Contract in docs/asyncapi.yaml
- Events der Nachbarn als Stubs
- Konventionen für Claude Code im Repository ablegen
- Konvention zum gemeinsamen Entwickeln festlegen (Feature Branches vs. alle auf `main` …)

Hausaufgabe bis 05.11.

- Contract fertig: docs/asyncapi.yaml im Service-Repo, passend zum Event-Katalog
- Service läuft allein
  - Happy-Path-Ausschnitt implementiert
  - lokale Tests implementiert und grün
  - Stubs für die Nachbar-Events angelegt
- Client, falls Ihr Team einen hat
  - Entwurf der Oberfläche
  - simuliertes Backend
  - Architekturentscheidung als ADR

Fragen?

archi-lab.io